**缓存数据的清理促发策略**

缓存数据的清理触发方案主要有两种：后台线程扫描，put时更新时清理。后台线程扫描，代码逻辑相对复杂，保证线程的存活性是一个难以解决的问题，有些托管环境不支持新建线程的；本地缓存的数据量相对较少，清理能很快进行，线程清理增加了线程的创建及调度的开销，这些开销的时间可能比缓存数据清理操作所需的时间还长。为保证本地缓存的灵活、简单、广泛、高效，本地缓存选择put时进行清理的策略。

在put时进行清理，又有以下三种实现方式

* 任一put时，发现过期就清理
* ~~任一put时，发现过期新建线程清理~~
* 两个put时，发现过期，只有一个进行清理操作。

Get会清理脏数据（避免阀值一直未达到，而张树筠）

这三种方式对比，明显最后一种方案在高并发时性能会更好。

本地缓存采用读账数据的方式，避免缓存穿透的问题。因此在清理时并非缓存过期就清除，而是在缓存过期一定时间后（如有效期的2倍），再进行过期操作。

失效前进行预加载， 触发预加载，减少账数据的读取。

失效并非控制。

Cache是基础

参考聚划算的

put 清理 vs 线程清理 vs 部分线程清理

十多个

提供页面级别的缓存组件、其它应用级别的缓存。（： pagecache的功能）

提前穿透，在缓存还有效时，就进行穿透。（时间点不好控制、差异性会比较大）

一致性分析： 适用于页面级别的缓存